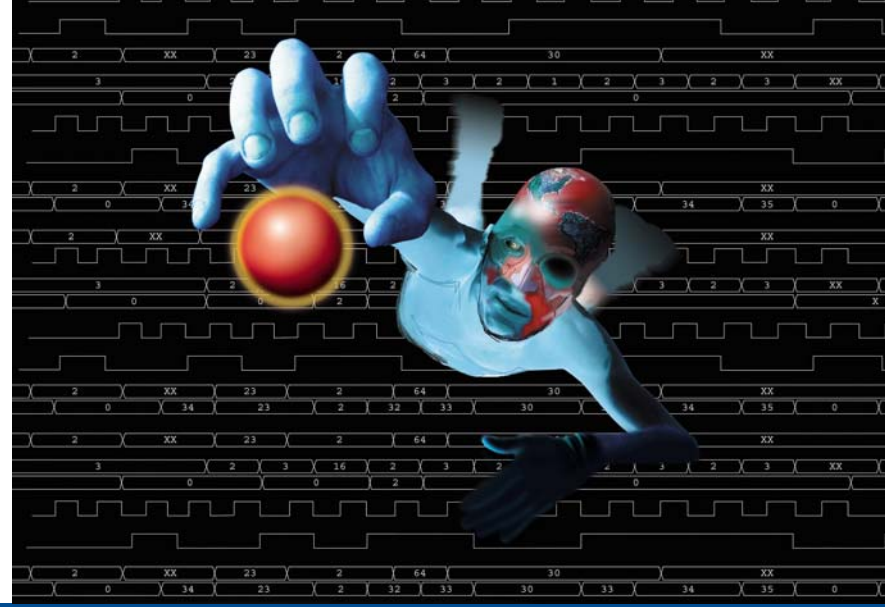
*Way Beyond Waveforms*



# Novas SystemVerilog Donation

Amadies Sun  Bassam Tabbara

NOVAS

# Outline

- Introduction and Objective
- SystemVerilog Today
- Our Proposal
- Donation
  - Reader
  - Writer
- Integration into SystemVerilog

NOVAS

# Introduction

- SystemVerilog
  - Design
  - Verification and Debug
    - Testbench
    - Coverage
    - Assertions
- Data from all these sources
- Many tools in each category

NOVAS

# Interfacing Objective

- Ease integration
  - Vendor
  - User
- Improve design productivity
  - Save costly data exchange
  - Innovative flows
- Users pick best-in-class tools, portable development
- Vendors compete via proprietary enhancements

NOVAS

# Available Schemes in SV 3.1

- Two primary means for interfacing
  - VPI
    - Design: Inherited from Verilog 2001
    - Coverage: Added in SV 3.1
    - Assertions: Added in SV 3.1
  - VCD
    - Mostly inherited from Verilog 95
    - Minor updates from Verilog 2001

NOVAS

# Pros and Cons of Scheme

- VCD
  - ☺ Accessible: Backward compatible
  - ☹ Arcane, inefficient, error-prone
- VPI
  - ☺ API level
  - ☺ Efficient mechanism: In-memory interfacing
- We think
  - VPI needs to be expanded anyway
  - VCD only serves as reference, no inherent value
  - We can easily “map” to existing VCD
  - We can always enhance VCD if need be later
  - **Focus on VPI for 3.1a**

NOVAS

# Our Proposal

- Enhance SV data dumping by extending VPI
  - Read/write dumping facilities donation from **Novas ffRead and ffWrite APIs of FSDB**
- VPI becomes extended API for data access tool interaction
  - In-memory or in file is hidden from user
  - All front-end tools would use same interface

NOVAS

# Technology Donation

# Reader (1 of 3)

- The dump file may contain two kinds of information:
  1) **Design hierarchies** which include dumped scopes and variables (and their data types) aka “tree” portion
  2) **Value changes** of the stored variables throughout the dump time ranges

NOVAS

# Reader (2 of 3)

- The Flow of Reading of the dump file is as follows:
  - Open the dump file
  - **Traverse the design hierarchies**
  - **Select variables we are interested in**
  - **Load variables' value changes onto memory**
  - **Create value-change traverse handle for variables**
  - ***Traverse the value changes via traverse functions***
  - Close the dump file

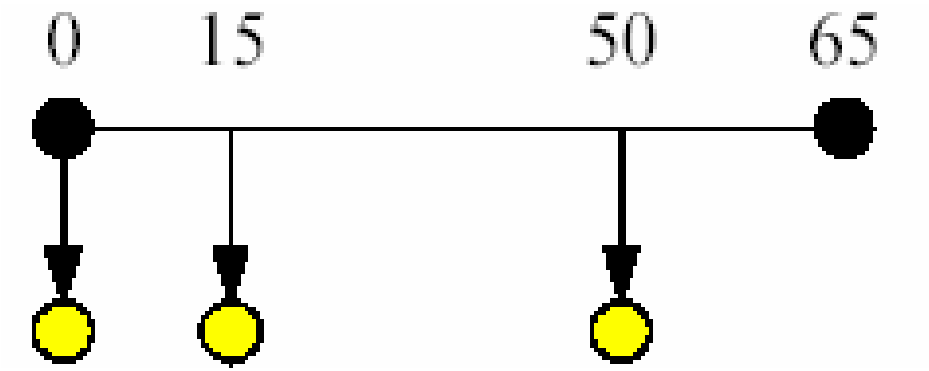
NOVAS

# Reader (3 of 3)

- Traverse functions
  - **ddrGotoXTag**: Move to the proper position, based on the time specified by the application
  - **ddrGotoPrevVC**: Move to the previous time and value change
  - **ddrGotoNextVC**: Move to the next time and value change
  - **ddrGetXTag**: Return the current time
  - **ddrGetVC**: Return the current value change
  - **ddrGetMinXTag**: Return the minimum time where value change occurs
  - **ddrGetMaxXTag**: Return the maximum time where value change occurs
  - **ddrHasIncoreVC**: Tell the application if this variable has value changes
  - **ddrFree**: Frees the data structures of the traverse handle

NOVAS

# Jump Behavior




- Jump to 10; return time is 0
- Jump to 15; return time is 15
- Jump to 65; return time is 50
- Jump to 30; return time is 15
- Jump to (-1); return time is 0
- Jump to 50; return time is 50

NOVAS

# Writer

- The value change creation is best done by **time-based scheme**:
  - Tag time creation
  - Value creation at that tag
- The time is created by calling `ddwCreateXCoorByHndl`
- The value is created by calling `ddwCreateVarValByHndl`

# Integrating into VPI

NOVAS

# How to Integrate into VPI

- Reader
  - A new set of **`#define`** would be added to accommodate the new types or object properties.
  - The VC traverse handle would be obtained appropriately from the original variable handle using **vpi_handle()** method.
  - The time traversal APIs would be integrated as additions to **vpi_control()** e.g. Next, Prev and so on.
  - The “get” APIs (value/current, min, or max time) would be added into the **vpi_get()**, and **vpi_get_value()** methods.
  - Dump object or file would be obtained from **$dumpvars, $readvars, …**
- Writer
  - **Value Change** callbacks of VPI

NOVAS

# Summary

- Industry needs a single, powerful, useful interface scheme among users/vendors
- Our proposal: Enhance VPI of SV 3.1
- Novas donation: Key elements from FSDB Reader/Writer API
- Goal: Work closely with the user/vendor members of SV (SV-CC) to solidify SV 3.1a with the data dump interface.

NOVAS